How to create a **Devcontainer** for your Python project 🐳

Jeroen Overschie

Machine Learning Engineer

GoDataDriven

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

So we need to align on:

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

So we need to align on:

- 📌 A specific Java version

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

So we need to align on:

- 📌 A specific Java version
- 📌 A specific Python version

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

So we need to align on:

- 📌 A specific Java version
- 📌 A specific Python version
- 📌 A specific `pyspark` version

The use case

You are assigned to setup a new repo for a team. The requirements are as follows:

So we need to align on:

- 📌 A specific Java version
- 📌 A specific Python version
- 📌 A specific `pyspark` version

→ otherwise we do not enjoy the guarantees we want in **production** code

PP

Petra the Python Dev

Hi my venv somehow got corrupted 🥲 . It's saying `No module named 'numpy'` even though I seemingly have it installed. Not sure how to fix.

P.S. this is also why I was not in the meeting.

PP **Petra the Python Dev**
Hi my venv somehow got corrupted 🥲. It's saying `No module named 'numpy'` even though I seemingly have it installed. Not sure how to fix.
P.S. this is also why I was not in the meeting.

EO **Erik the old project maintainer**
Hey so I was working on maintaining the old project. Just found out that some of the dependencies don't even work on my MacOS version anymore. FML

PP

Petra the Python Dev
Hi my venv somehow got corrupted 🥲. It's saying `No module named 'numpy'` even though I seemingly have it installed. Not sure how to fix.
P.S. this is also why I was not in the meeting.

BW

Billy Windows
I'm trying to dockerize the project but it's haaard. Everything that works on my Windows laptop seems to fail on Linux.

EO

Erik the old project maintainer
Hey so I was working on maintaining the old project. Just found out that some of the dependencies don't even work on my MacOS version anymore. FML

PP **Petra the Python Dev**
Hi my venv somehow got corrupted 🥲. It's saying `No module named 'numpy'` even though I seemingly have it installed. Not sure how to fix.
P.S. this is also why I was not in the meeting.

EO **Erik the old project maintainer**
Hey so I was working on maintaining the old project. Just found out that some of the dependencies don't even work on my MacOS version anymore. FML

BW **Billy Windows**
I'm trying to dockerize the project but it's haaard. Everything that works on my Windows laptop seems to fail on Linux.

JJ **Johnny Junior**
Hii. Can you maybe run `pip show pyspark`? I'm curious which pyspark version you are running 🧐. Because if it works for you but not for me and also not in the CI maybe your environment is different than both. Just checking.

PP **Petra the Python Dev**
Hi my venv somehow got corrupted 🥲. It's saying `No module named 'numpy'` even though I seemingly have it installed. Not sure how to fix.
P.S. this is also why I was not in the meeting.

EO **Erik the old project maintainer**
Hey so I was working on maintaining the old project. Just found out that some of the dependencies don't even work on my MacOS version anymore. FML

BW **Billy Windows**
I'm trying to dockerize the project but it's haaard. Everything that works on my Windows laptop seems to fail on Linux.

JJ **Johnny Junior**
Hii. Can you maybe run `pip show pyspark`? I'm curious which pyspark version you are running 🧐. Because if it works for you but not for me and also not in the CI maybe your environment is different than both. Just checking.

JJ **Johnny Junior**
Good day. I'm new at the company and wanted to get started working on the repo. Tried following the README steps but doesn't work. By any chance: are there any more detailed docs available for project setup? No right? 🙃

🐛 Issues

🐛 Issues

Petra the Python Dev

Corrupted Virtual Environment

🐛 Issues

PP	Petra the Python Dev	Corrupted Virtual Environment
EO	Erik the old project maintainer	Outdated project dependencies

🐛 Issues

PP Petra the Python Dev	Corrupted Virtual Environment
EO Erik the old project maintainer	Outdated project dependencies
BW Billy Windows	Containerisation & going into production e.g. Windows / Linux / MacOS

🪲 Issues

PP Petra the Python Dev	Corrupted Virtual Environment
EO Erik the old project maintainer	Outdated project dependencies
BW Billy Windows	Containerisation & going into production e.g. Windows / Linux / MacOS
PU PySpark user	Inconsistent environments e.g. local ↔ CI ↔ prod ↔ team members

🐛 Issues

PP Petra the Python Dev	Corrupted Virtual Environment
EO Erik the old project maintainer	Outdated project dependencies
BW Billy Windows	Containerisation & going into production e.g. Windows / Linux / MacOS
PU PySpark user	Inconsistent environments e.g. local ↔ CI ↔ prod ↔ team members
JJ Johnny Junior	No formal specification of install steps and missing docs 📚

🐛 Issues

PP	Petra the Python Dev	Corrupted Virtual Environment
EO	Erik the old project maintainer	Outdated project dependencies
BW	Billy Windows	Containerisation & going into production e.g. Windows / Linux / MacOS
PU	PySpark user	Inconsistent environments e.g. local ↔ CI ↔ prod ↔ team members
JJ	Johnny Junior	No formal specification of install steps and missing docs 📚

🔄 How to get a **reproducible** Dev environment?

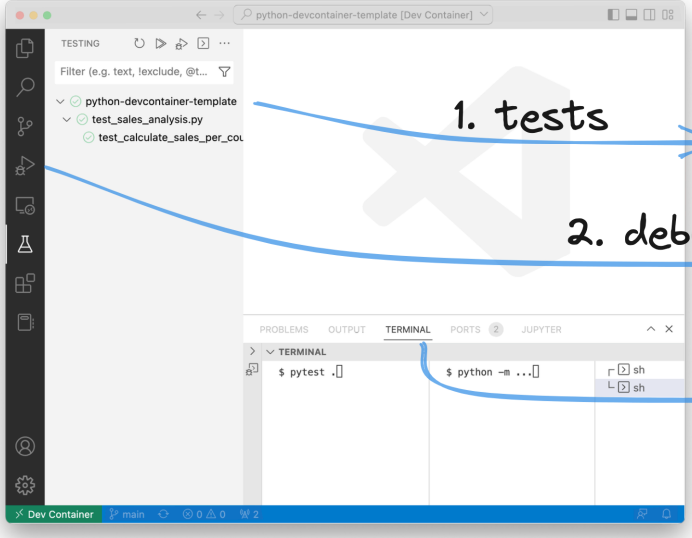
Devcontainers to the rescue  !

Devcontainers to the rescue ⛑️!

🐋 **Docker** helps us create a formal definition of our environment.

🔌 **Devcontainers** allow you to connect your editor (IDE) to that container.

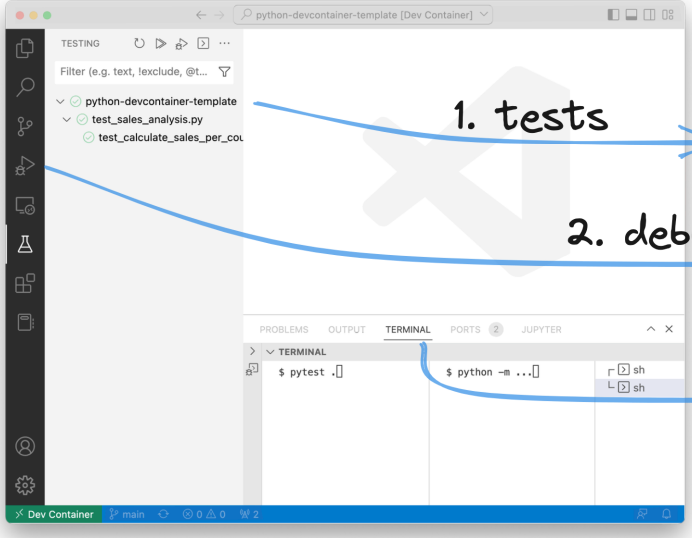
Develop in your IDE,
but run ...



... inside your
Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

Develop in your IDE,
but run ...

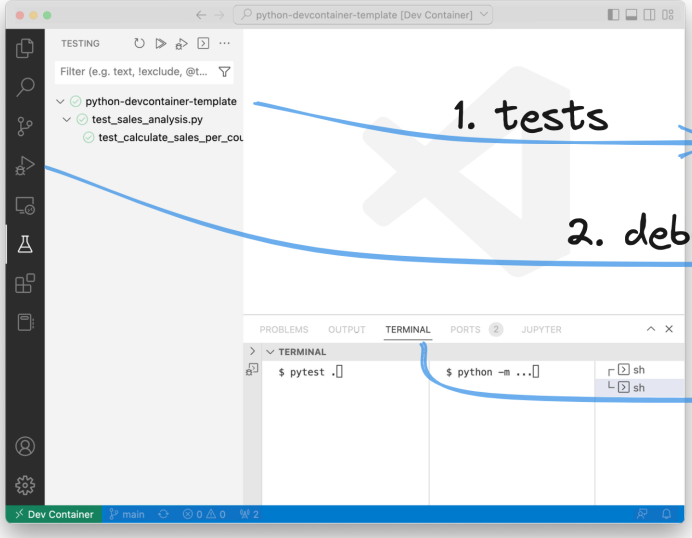


... inside your
Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

🔄 **Reproducible** means:

Develop in your IDE,
but run ...



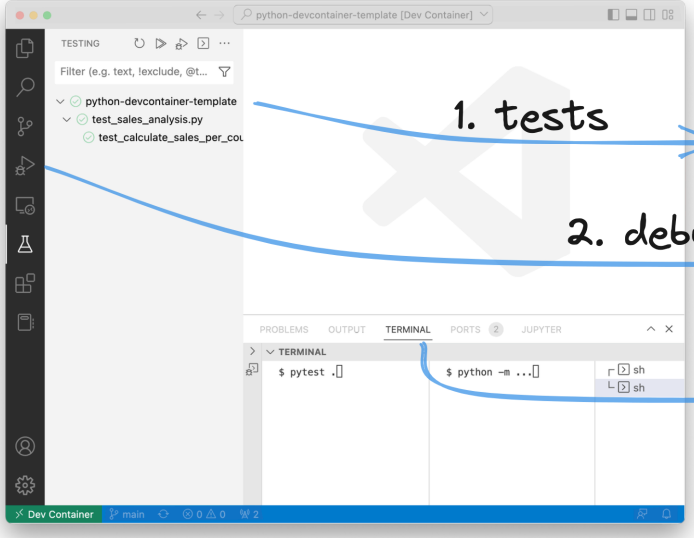
... inside your
Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

🔄 **Reproducible** means:

- ⚡ Faster onboarding

Develop in your IDE,
but run ...



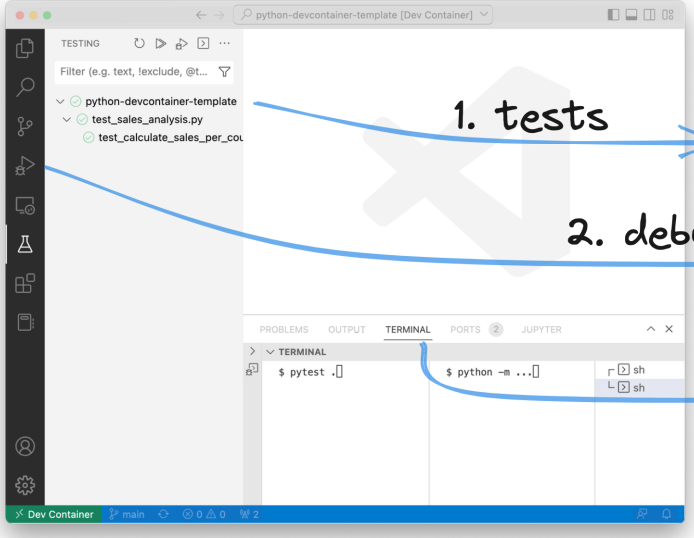
... inside your
Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

🔄 **Reproducible** means:

- ⚡ Faster onboarding
- 👫 Better alignment between team members

Develop in your IDE,
but run ...



1. tests

2. debugging

3. terminal

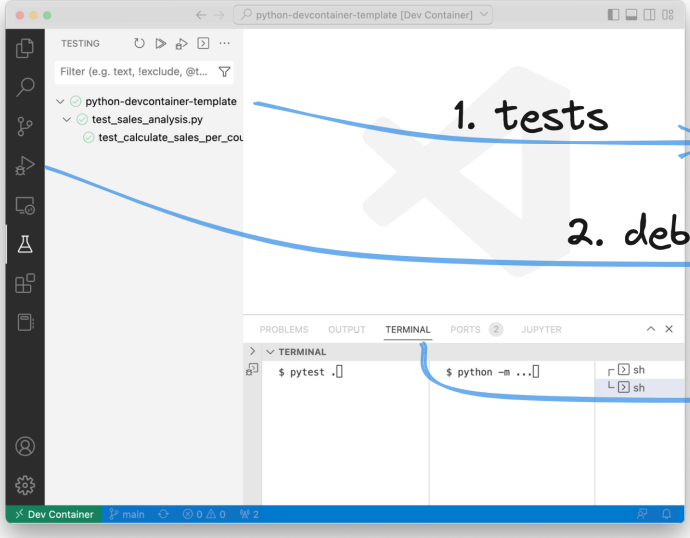
... inside your
Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

🔄 **Reproducible** means:

- ⚡ Faster onboarding
- 👫 Better alignment between team members
- ⏱️ Smaller gap to production

Develop in your IDE,
but run ...



... inside your Docker container

- ✅ reproducibility
- ✅ isolation
- ✅ native experience

🔄 **Reproducible** means:

- ⚡ Faster onboarding
- 👫 Better alignment between team members
- ⏱️ Smaller gap to production

Downsides?

- Docker knowledge

 Let’s build a Devcontainer!

Let's say we have a really simple project that looks like this:

```
$ tree .
.
├── README.md
├── requirements.txt
├── requirements-dev.txt
├── sales_analysis.py
└── test_sales_analysis.py
```

The `.devcontainer` folder

Your Devcontainer spec will live inside the `.devcontainer` folder.

The `.devcontainer` folder

Your Devcontainer spec will live inside the `.devcontainer` folder.

There will be two main files:

- `devcontainer.json`
- `Dockerfile`

Create a new file called `devcontainer.json`:

```
{
    "build": {
        "dockerfile": "Dockerfile",
        "context": ".."
    }
}
```

So how does this `Dockerfile` look like?

```
1  FROM python:3.10
2
3  # Install Java
4  RUN apt update && \
5      apt install -y sudo && \
6      sudo apt install default-jdk -y
7
8  ## Pip dependencies
9  # Upgrade pip
10 RUN pip install --upgrade pip
11 # Install production dependencies
12 COPY requirements.txt /tmp/requirements.txt
13 RUN pip install -r /tmp/requirements.txt && \
14     rm /tmp/requirements.txt
15 # Install development dependencies
16 COPY requirements-dev.txt /tmp/requirements-dev.txt
17 RUN pip install -r /tmp/requirements-dev.txt && \
18     rm /tmp/requirements-dev.txt
```

So how does this `Dockerfile` look like?

```
FROM python:3.10

# Install Java
RUN apt update && \
    apt install -y sudo && \
    sudo apt install default-jdk -y

## Pip dependencies
# Upgrade pip
RUN pip install --upgrade pip
# Install production dependencies
COPY requirements.txt /tmp/requirements.txt
RUN pip install -r /tmp/requirements.txt && \
    rm /tmp/requirements.txt
# Install development dependencies
COPY requirements-dev.txt /tmp/requirements-dev.txt
RUN pip install -r /tmp/requirements-dev.txt && \
    rm /tmp/requirements-dev.txt
```

So how does this `Dockerfile` look like?

```
1  FROM python:3.10
2
3  # Install Java
4  RUN apt update && \
5      apt install -y sudo && \
6      sudo apt install default-jdk -y
7
8  ## Pip dependencies
9  # Upgrade pip
10 RUN pip install --upgrade pip
11 # Install production dependencies
12 COPY requirements.txt /tmp/requirements.txt
13 RUN pip install -r /tmp/requirements.txt && \
14     rm /tmp/requirements.txt
15 # Install development dependencies
16 COPY requirements-dev.txt /tmp/requirements-dev.txt
17 RUN pip install -r /tmp/requirements-dev.txt && \
18     rm /tmp/requirements-dev.txt
```

So how does this `Dockerfile` look like?

```
1  FROM python:3.10
2
3  # Install Java
4  RUN apt update && \
5      apt install -y sudo && \
6      sudo apt install default-jdk -y
7
8  ## Pip dependencies
9  # Upgrade pip
10 RUN pip install --upgrade pip
11 # Install production dependencies
12 COPY requirements.txt /tmp/requirements.txt
13 RUN pip install -r /tmp/requirements.txt && \
14     rm /tmp/requirements.txt
15 # Install development dependencies
16 COPY requirements-dev.txt /tmp/requirements-dev.txt
17 RUN pip install -r /tmp/requirements-dev.txt && \
18     rm /tmp/requirements-dev.txt
```

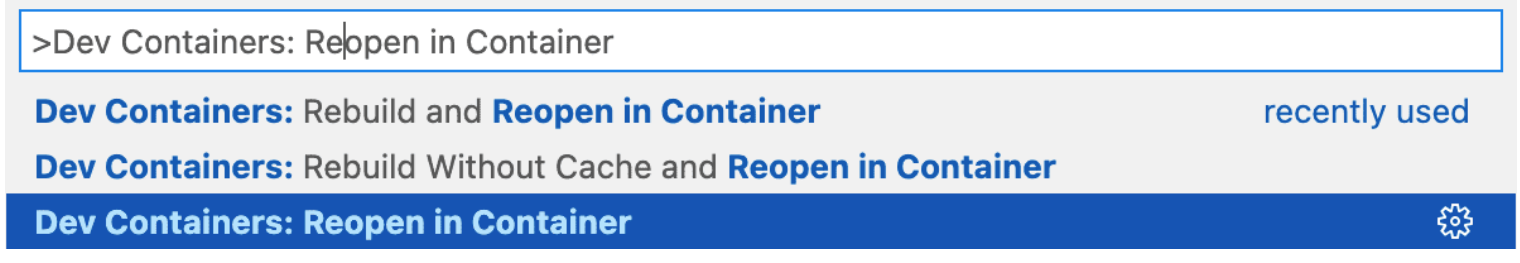
So how does this `Dockerfile` look like?

```
1  FROM python:3.10
2
3  # Install Java
4  RUN apt update && \
5      apt install -y sudo && \
6      sudo apt install default-jdk -y
7
8  ## Pip dependencies
9  # Upgrade pip
10 RUN pip install --upgrade pip
11 # Install production dependencies
12 COPY requirements.txt /tmp/requirements.txt
13 RUN pip install -r /tmp/requirements.txt && \
14     rm /tmp/requirements.txt
15 # Install development dependencies
16 COPY requirements-dev.txt /tmp/requirements-dev.txt
17 RUN pip install -r /tmp/requirements-dev.txt && \
18     rm /tmp/requirements-dev.txt
```

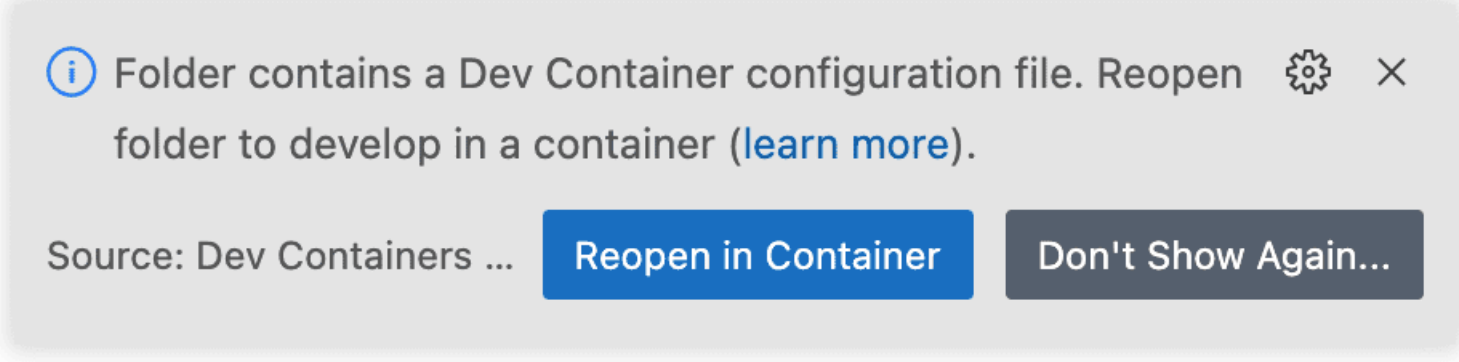
Opening the Devcontainer

The `.devcontainer` folder in place, now it's time to open our Devcontainer.

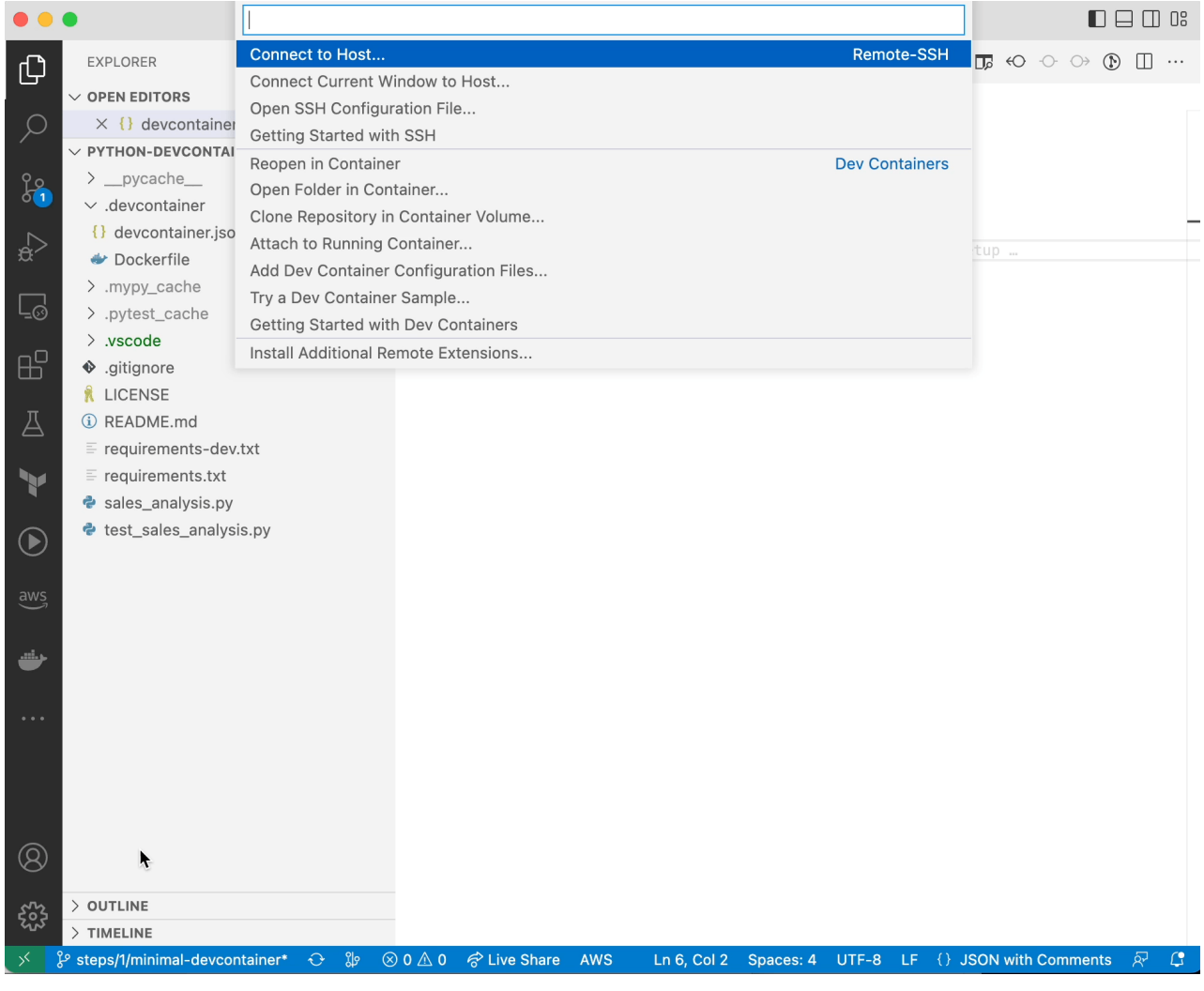
Open up the command pallete (`CMD` + `Shift` + `P`) and select "*Dev Containers: Reopen in Container*":



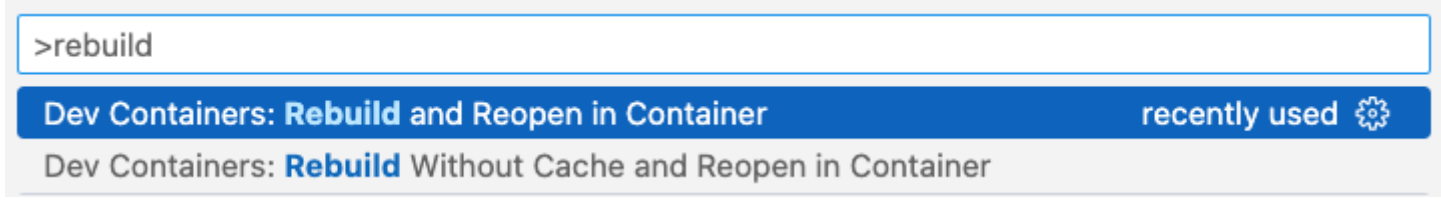
Upon opening a repo with a valid `.devcontainer` folder, you are already notified:



Your VSCode is now connected to the Docker container 🙌:



Rebuilding allows you to get a fresh environment anytime you want:



What is happening under the hood 🚗

Besides starting the Docker image and attaching the terminal to it, VSCode is doing a couple more things:

What is happening under the hood 🚗

Besides starting the Docker image and attaching the terminal to it, VSCode is doing a couple more things:

1. **VSCode Server** is being installed on your Devcontainer.

   VSCode Server is installed as a service in the container itself so your VSCode installation can communicate with the container.

   For example, install and run **extensions**.

What is happening under the hood 🚗

Besides starting the Docker image and attaching the terminal to it, VSCode is doing a couple more things:

1. **VSCode Server** is being installed on your Devcontainer.

   VSCode Server is installed as a service in the container itself so your VSCode installation can communicate with the container.

   For example, install and run **extensions**.

2. **Config is copied** over.

   Config like `~/.gitconfig` and `~/.ssh/known_hosts` are copied over to their respective locations in the container.

What is happening under the hood 🚗

Besides starting the Docker image and attaching the terminal to it, VSCode is doing a couple more things:

1. **VSCode Server** is being installed on your Devcontainer.

   VSCode Server is installed as a service in the container itself so your VSCode installation can communicate with the container.

   For example, install and run **extensions**.

2. **Config is copied** over.

   Config like `~/.gitconfig` and `~/.ssh/known_hosts` are copied over to their respective locations in the container.

3. **Filesystem mounts**.

   VSCode automatically takes care of mounting:

   - The folder you are running the Devcontainer from.
   - Your VSCode workspace folder.

Opening your Devcontainer with the click of a button

Your entire project setup is now encapsulated in the Devcontainer. So actually we can add a **Markdown** button to open up the Devcontainer:

Opening your Devcontainer with the click of a button

Your entire project setup is now encapsulated in the Devcontainer. So actually we can add a **Markdown** button to open up the Devcontainer:

```
[
    ![Open in Remote - Containers](
        https://img.shields.io/static/v1?label=Remote%20-%20Containers&message=Open&color=blue&logo=visualstudioco
    )
](
    https://vscode.dev/redirect?url=vscode://ms-vscode-remote.remote-containers/cloneInVolume?url=https://github.com/godatadriven/python-devcontainer-template
)
```

Opening your Devcontainer with the click of a button

Your entire project setup is now encapsulated in the Devcontainer. So actually we can add a **Markdown** button to open up the Devcontainer:

```
[
    ![Open in Remote - Containers](
        https://img.shields.io/static/v1?label=Remote%20-%20Containers&message=Open&color=blue&logo=visualstudiocod
    )
](
    https://vscode.dev/redirect?url=vscode://ms-vscode-remote.remote-containers/cloneInVolume?url=https://github.com/godatadriven/python-devcontainer-template
)
```

Which basically means, open this URL:

```
vscode://
  ms-vscode-remote.remote-containers/
  cloneInVolume?
  url=https://github.com/godatadriven/python-
devcontainer-template
```

Just modify the GitHub URL after `url=` ✓.

This renders the following button:

What kind of README would you rather like?

49 lines (29 sloc) | 932 Bytes — Raw | Blame

Your fantastic repo

Hi! Welcome to the team ! Let's get you started quickly. This is what you have to do.

Installation

1. Install Java 11.0.16
2. Install Python 3.10.8

Virtual environment

There are a couple ways to go about this.

1. `pyenv`

   First, install pyenv using the pyenv-installer.

   Then, create a new environment using:

   ```
   pyenv shell 3.10.8
   pyenv virtualenv my-venv
   pyenv shell my-venv
   ```

2. Conda

   Install conda and create a virtualenv.

3. Python venv

   ```
   python -m venv venv
   source ./venv/bin/activate
   ```

pip packages

1. Make sure to `pip` install the following packages:
   - black
   - pytest
   - jupyter (not necessary when installing for CI or prod)
   - mypy

⚠️ Note: this is entire setup is known to work for Debian GNU/Linux 11 (bullseye), not tested for other Operating Systems.

15 lines (8 sloc) | 1 KB — Raw | Blame

Your fantastic repo

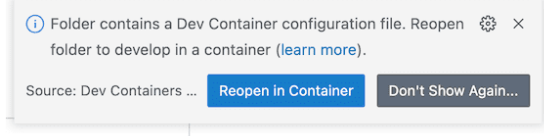
If you happen to use VSCode as your editor, you can open the repo in a **Devcontainer**. Devcontainers allow you to develop *inside* a Docker container - which means all dependencies and packages are automatically set up for you. First, make sure you have the Remote Development extension installed.

Then, you can do two things.

1. Click the following button:

2. Or, open up the repo in VSCode. Then, you should see the following notification:



That's it 🙌 Enjoy developing.

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

- Install a **non-root user** for extra safety and good-practice

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

- Install a **non-root user** for extra safety and good-practice
- Pass in **custom VSCode settings** and install extensions by default

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

- Install a **non-root user** for extra safety and good-practice
- Pass in **custom VSCode settings** and install extensions by default
- Be able to access Spark UI (**opening up port 4040**)

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

- Install a **non-root user** for extra safety and good-practice
- Pass in **custom VSCode settings** and install extensions by default
- Be able to access Spark UI (**opening up port 4040**)
- Run **Continuous Integration** (CI) in the Devcontainer

Extending the Devcontainer

We have built a working Devcontainer, that is great! But a couple things are still missing.

- Install a **non-root user** for extra safety and good-practice
- Pass in **custom VSCode settings** and install extensions by default
- Be able to access Spark UI (**opening up port 4040**)
- Run **Continuous Integration** (CI) in the Devcontainer

Let's see how.

Installing a non-root user

If you `pip install` a new package, you will see the following message:

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment instead: https://pip.pypa.io/warnings/venv
```

So let's go ahead and create a user for this scenario.

```
# Add non-root user
ARG USERNAME=nonroot
RUN groupadd --gid 1000 $USERNAME && \
    useradd --uid 1000 --gid 1000 -m $USERNAME
## Make sure to reflect new user in PATH
ENV PATH="/home/${USERNAME}/.local/bin:${PATH}"
USER $USERNAME
```

So let's go ahead and create a user for this scenario.

```
# Add non-root user
ARG USERNAME=nonroot
RUN groupadd --gid 1000 $USERNAME && \
    useradd --uid 1000 --gid 1000 -m $USERNAME
## Make sure to reflect new user in PATH
ENV PATH="/home/${USERNAME}/.local/bin:${PATH}"
USER $USERNAME
```

Add the following property to `devcontainer.json`:

```
"remoteUser": "nonroot"
```

So let's go ahead and create a user for this scenario.

```
# Add non-root user
ARG USERNAME=nonroot
RUN groupadd --gid 1000 $USERNAME && \
    useradd --uid 1000 --gid 1000 -m $USERNAME
## Make sure to reflect new user in PATH
ENV PATH="/home/${USERNAME}/.local/bin:${PATH}"
USER $USERNAME
```

Add the following property to `devcontainer.json`:

```
"remoteUser": "nonroot"
```

That's great! When we now start the container we should connect as the user `nonroot`.

Passing custom VSCode settings

Passing custom VSCode settings

```
"customizations": {
    "vscode": {
        "extensions": [
            "ms-python.python"
        ],
        "settings": {
            "python.testing.pytestArgs": [
                "."
            ],
            "python.testing.unittestEnabled": false,
            "python.testing.pytestEnabled": true,
            "python.formatting.provider": "black",
            "python.linting.mypyEnabled": true,
            "python.linting.enabled": true
        }
    }
}
```

Accessing Spark UI

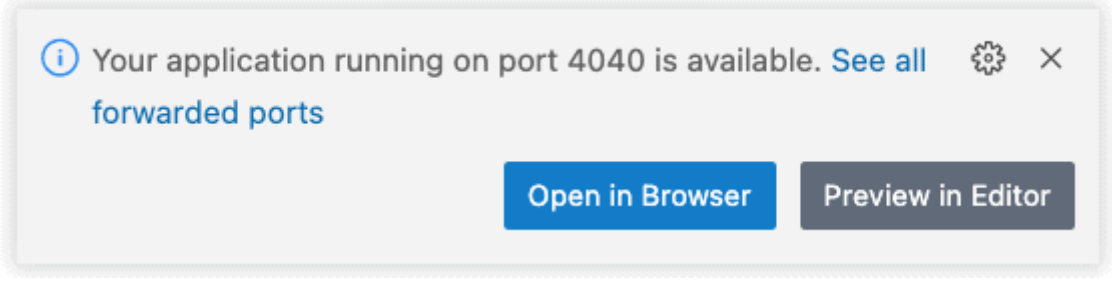
Since we are using pyspark, it would be nice to be able to access **Spark UI**.

Accessing Spark UI

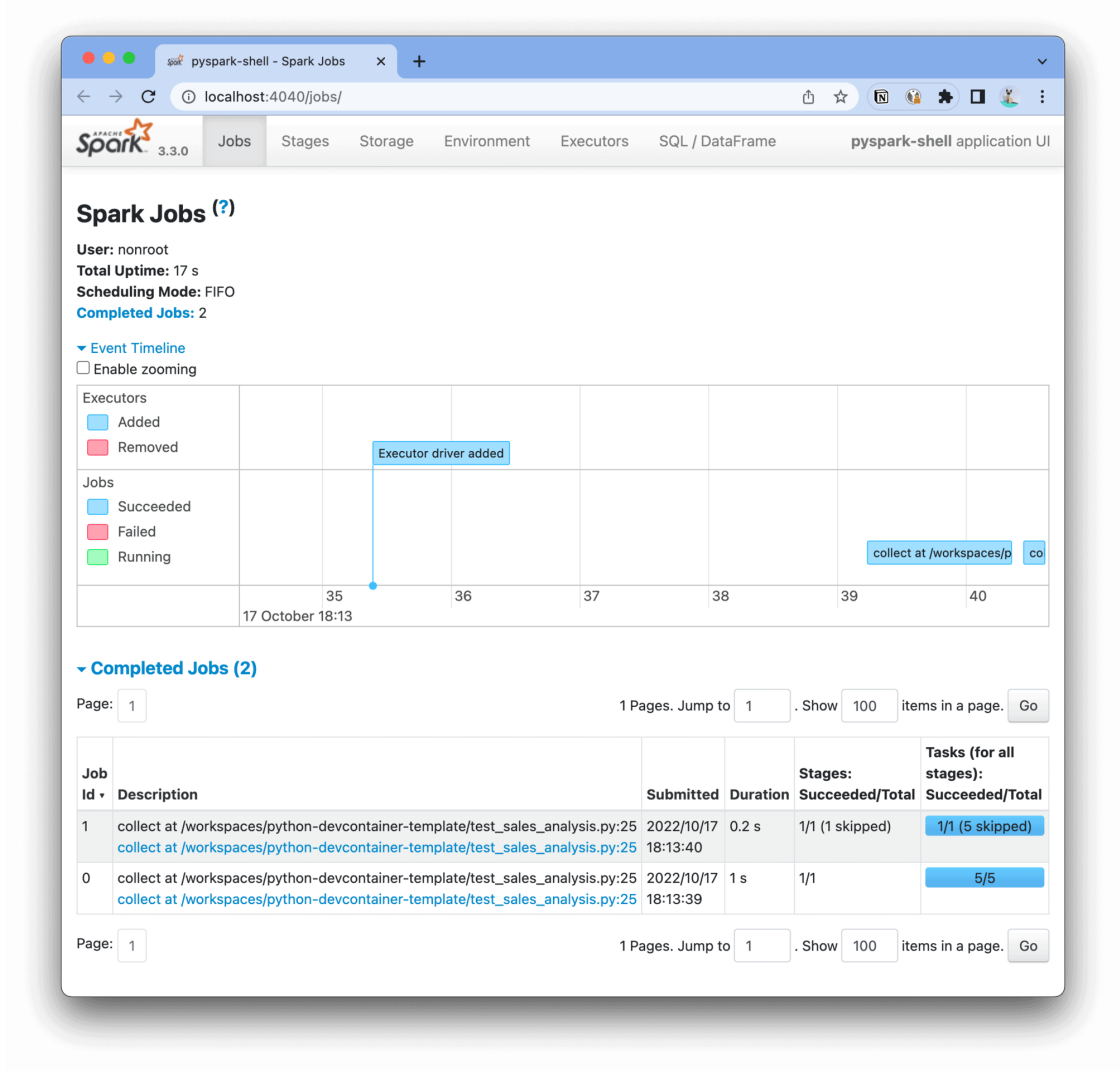
Since we are using pyspark, it would be nice to be able to access **Spark UI**.

```
"portsAttributes": {
    "4040": {
        "label": "SparkUI",
        "onAutoForward": "notify"
    }
},

"forwardPorts": [
    4040
]
```

When we now run our code, we get a notification we can open Spark UI in the browser:



Resulting in the Spark UI like we know it:



Running our CI in the Devcontainer

Running our CI in the Devcontainer

There are two basic options:

1. Build the Docker image *within* the CI/CD pipeline
2. Prebuilding the image

Let's see about option number (1).

1. Build the Docker image *within* the CI/CD pipeline

Luckily, a GitHub Action was already setup for us to do exactly this:

devcontainers/ci

To now build, push and run a command in the Devcontainer is as easy as:

```
1
2  name: Python app
3
4  on:
5    ...
6
7  jobs:
8   build:
9    runs-on: ubuntu-latest
10
11   steps:
12    - name: Checkout (GitHub)
13     uses: actions/checkout@v3
14
15    - name: Login to GitHub Container Registry
16     uses: docker/login-action@v2
17     with:
18      registry: ghcr.io
```

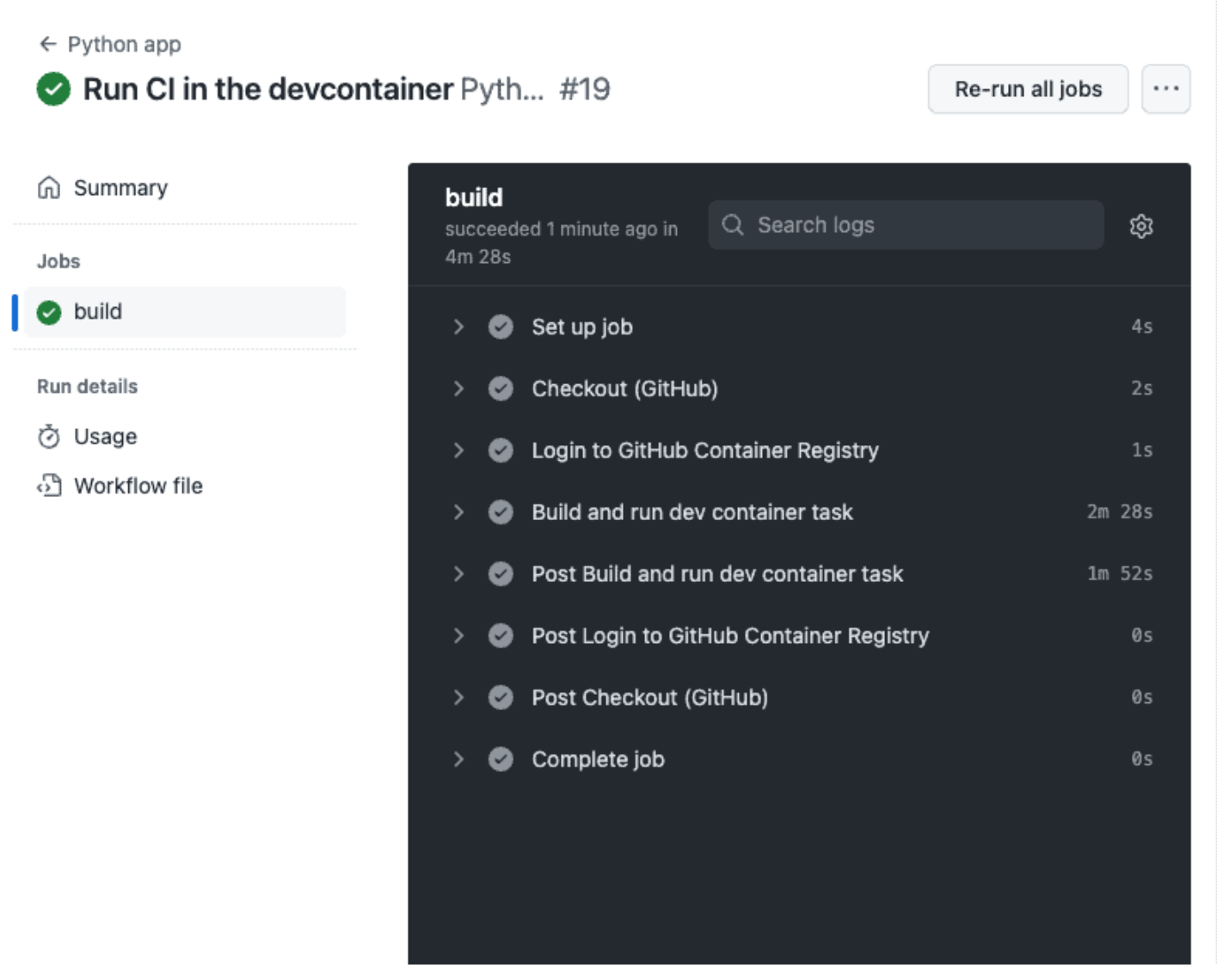
To now build, push and run a command in the Devcontainer is as easy as:

```yaml
    runs-on: ubuntu-latest

    steps:
      - name: Checkout (GitHub)
        uses: actions/checkout@v3

      - name: Login to GitHub Container Registry
        uses: docker/login-action@v2
        with:
          registry: ghcr.io
          username: ${{ github.repository_owner }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Build and run dev container task
        uses: devcontainers/ci@v0.2
        with:
          imageName: ghcr.io/${{ github.repository }}/devcontainer
```

To now build, push and run a command in the Devcontainer is as easy as:

```
10
11    steps:
12     - name: Checkout (GitHub)
13       uses: actions/checkout@v3
14
15     - name: Login to GitHub Container Registry
16       uses: docker/login-action@v2
17       with:
18         registry: ghcr.io
19         username: ${{ github.repository_owner }}
20         password: ${{ secrets.GITHUB_TOKEN }}
21
22     - name: Build and run dev container task
23       uses: devcontainers/ci@v0.2
24       with:
25         imageName: ghcr.io/${{ github.repository
   }}/devcontainer
26         runCmd: pytest .
```

See below a trace of the executed GitHub Action:



Awesome!

The final Devcontainer definition

We built the following Devcontainer definitions.

First, `devcontainer.json`:

```
{
    "build": {
        "dockerfile": "Dockerfile",
        "context": ".."
    },

    "remoteUser": "nonroot",

    "customizations": {
        "vscode": {
            "extensions": [
                "ms-python.python"
            ],
            "settings": {
                "python.testing.pytestArgs": [
                    "."
                ],
                "python.testing.unittestEnabled": false,
                "python.testing.pytestEnabled": true,
                "python.formatting.provider": "black",
                "python.linting.mypyEnabled": true,
                "python.linting.enabled": true
            }
        }
    },
    ...
}
```

```
"portsAttributes": {
    "4040": {
        "label": "SparkUI",
        "onAutoForward": "notify"
    }
},

"forwardPorts": [
    4040
]
```

And our `Dockerfile`:

```
FROM python:3.10

# Install Java
RUN apt update && \
    apt install -y sudo && \
    sudo apt install default-jdk -y

# Add non-root user
ARG USERNAME=nonroot
RUN groupadd --gid 1000 $USERNAME && \
    useradd --uid 1000 --gid 1000 -m $USERNAME
## Make sure to reflect new user in PATH
ENV PATH="/home/${USERNAME}/.local/bin:${PATH}"
USER $USERNAME

## Pip dependencies
# Upgrade pip
RUN pip install --upgrade pip
# Install production dependencies
COPY --chown=nonroot:1000 requirements.txt /tmp/requirements.txt
RUN pip install -r /tmp/requirements.txt && \
    rm /tmp/requirements.txt
# Install development dependencies
COPY --chown=nonroot:1000 requirements-dev.txt /tmp/requirements-
dev.txt
RUN pip install -r /tmp/requirements-dev.txt && \
    rm /tmp/requirements-dev.txt
```

Going further 🔮

Going further 🔮

- Mounting directories

  💡 Pro tip: mount your AWS/GCP/Azure credentials

Going further 🔮

- Mounting directories

  💡 Pro tip: mount your AWS/GCP/Azure credentials

- Devcontainer templates

Going further 🔮

- Mounting directories

  💡 Pro tip: mount your AWS/GCP/Azure credentials

- Devcontainer templates
- Devcontainer features

Going further 🔮

- Mounting directories

  💡 Pro tip: mount your AWS/GCP/Azure credentials

- Devcontainer templates
- Devcontainer features

- **Remote Development** 🌐

  e.g. GitHub Codespaces, VM's on Azure/GCP/AWS

Going further 🔮

- Mounting directories

  💡 Pro tip: mount your AWS/GCP/Azure credentials

- Devcontainer templates
- Devcontainer features
- **Remote Development** 🌐

  e.g. GitHub Codespaces, VM's on Azure/GCP/AWS

... and much more (see references slide)

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

🔄 **Reproducible** means:

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

🔄 **Reproducible** means:

- ⚡ Faster onboarding

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

🔄 **Reproducible** means:

- ⚡ Faster onboarding
- 👫 Better alignment between team members

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

🔄 **Reproducible** means:

- ⚡ Faster onboarding
- 👫 Better alignment between team members
- ⏱️ Smaller gap to production

Concluding

🔌 **Devcontainers** connect your IDE to a running 🐋 Docker container.

→ *reproducibility* & *isolation* whilst getting a *native* experience.

🔄 **Reproducible** means:

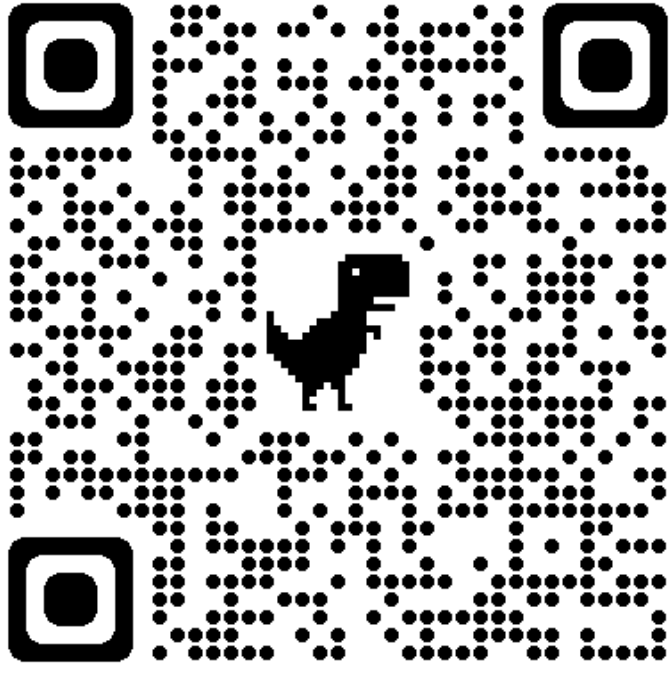
- ⚡ Faster onboarding
- 👫 Better alignment between team members
- ⏱️ Smaller gap to production

Now only VSCode, but open specification taking shape.


PP
EO
BW
PU
JJ


PP
EO
BW
PU
JJ

Thanks! 🙌🏻



github.com/godatadriven/python-devcontainer-template

Awesome resources

Associated blog post:

- How to create a Devcontainer for your Python project 🐋

Spec:

- containers.dev. The official Devcontainer specification.
    - Devcontainer templates
    - Devcontainer features

Docs:

- Dev Containers VSCode extension. The extension required to connect VSCode to a Devcontainer.
- Mounting file directories in Devcontainers.
- Add a non-root user to a container. More explanations & instructions for adding a non-root user to your `Dockerfile` and `devcontainer.json`.
- Pre-building dev container images

Repo's:

- github.com/devcontainers/ci. Run your CI in your Devcontainers. Built on the Devcontainer CLI.
- github.com/devcontainers/images. A collection of ready-to-use Devcontainer images.
- github.com/manekinekko/awesome-devcontainers. A repo pointing to yet even more awesome resources.